Mở đầu chương trình

```
.MODEL SMALL
.STACK 100H
.DATA
;các định nghĩa số liệu ở đây
```



```
.CODE
MAIN    PROC
;các lệnh ở đây
MAIN    ENDP
;các thủ tục khác ở đây
END     MAIN
```

Int 21H đọc kí tư, hiển thị kí tự, xuống dòng

```
TITLE   PGM4_1:SAMPLE PROGRAM
.MODEL  SMALL
.STACK  100H
.CODE
MAIN    PROC
```



```
        MOV     AH,2            ; hàm hiển thị ký tự
        MOV     DL,'?'          ; ký tự là "?"
        INT     21h             ; hiển thị ký tự
;vào một ký tự
        MOV     AH,1            ; hàm đọc một ký tự
        INT     21h             ; ký tự trong AL
        MOV     BL,AL           ; cất ký tự trong BL
;xuống dòng mới:
        MOV     AH,2            ; hàm hiển thị ký tự
        MOV     DL,0Dh          ; về đầu dòng
        INT     21h             ; thực hiện về đầu dòng
        MOV     DL,0Ah          ; xuống dòng
        INT     21h             ; thực hiện xuống dòng
;hiển thị ký tự:
        MOV     DL,BL           ;lấy ký tự
        INT     21h             ;và hiển thị nó
;trở về DOS
        MOV     AH,4CH          ;hàm thoát về DOS
        INT     21H             ;thoát về DOS
MAIN    ENDP
        END     MAIN
```

In ra 1 chuoi kí tự

Chương trình PGM4_2.ASM

```
TITLE    PGM4_2:   Chương trình in chuỗi ký tự.
.MODEL          SMALL
.STACK          100H
.DATA
MSG     DB      "HELLO!$"
.CODE
MAIN    PROC
;khởi tạo DS
        MOV     AX,@DATA
        MOV     DS,AX
;hiển thị thông báo
        LEA     DX,MSG          ; lấy thông báo
        MOV     AH,9            ; hàm hiển thị chuỗi
        INT     21h             ; hiển thị chuỗi
;trở về DOS
        MOV     AH,4CH
        INT     21H
MAIN    ENDP
        END     MAIN
```

```
.MODEL  SMALL
.STACK  100H
.DATA
CR      EQU     0DH
LF      EQU     0AH
MSG1    DB      'ENTER  A  LOWER CASE  LETTER : $ '
MSG2    DB      CR,LF,' IN  UPPER  CASE IT IS :'
CHAR    DB      ?,' $ '
.CODE
MAIN    PROC
;khởi tạo DS
        MOV     AX,@DATA
        MOV     DS,AX
;in dòng nhắc người sử dụng
        LEA     DX,MSG1   ;lấy thông báo đầu tiên
        MOV     AH,9      ;hàm hiển thị chuỗi
        INT     21h       ;hiển thị thông báo đầu tiên
;vào một ký tự và đổi thành chữ hoa
        MOV     AH,1      ;hàm đọc một ký tự
        INT     21h       ;đọc một chữ thường vào AL
        SUB     AL,20H    ;đổi thành chữ hoa
        MOV     CHAR,AL   ;và lưu trữ nó
;hiển thị trên dòng tiếp theo
        LEA     DX,MSG2         ;lấy thông báo thứ hai
        MOV     AH,9            ;hàm hiển thị chuỗi
        INT     21h             ;hiển thị thông báo thứ
                                ;hai và chữ hoa
;trở về DOS
        MOV     AH,4CH
        INT     21H
MAIN    ENDP
        END     MAIN
```

Chuyển chữ thường thanhf chữ Hoa

Lệnh nhảy

Ký hiệu	Chức năng	Điều kiện nhảy
JG/JNLE	nhảy nếu lớn hơn nhảy nếu không nhỏ hơn hay bằng	ZF=0 và SF=OF
JGE/JNL	nhảy nếu lớn hơn hay bằng nhảy nếu không nhỏ hơn	SF=OF
JL/JNGE	nhảy nếu nhỏ hơn nhảy nếu không lớn hơn hay bằng	SF<>OF
JLE/JNG	nhảy nếu nhỏ hơn hay bằng nhảy nếu không lớn hơn	ZF=1 hay SF=OF

Các lệnh nhảy không dấu.

Ký hiệu	Chức năng	Điều kiện nhảy
JA/JNBE	nhảy nếu lớn hơn nhảy nếu không nhỏ hơn hay bằng	CF=0 và ZF=0
JAE/JNB	nhảy nếu lớn hơn hay bằng nhảy nếu không nhỏ hơn	CF=0
JB/JNAE	nhảy nếu nhỏ hơn nhảy nếu không lớn hơn hay bằng	CF=1

JBE/JNA	nhảy nếu nhỏ hơn hay bằng nhảy nếu không lớn hơn	CF=1 hay ZF=1

Các lệnh nhảy điều kiện đơn.

Ký hiệu	Chức năng	Điều kiện nhảy
JE/JZ	nhảy nếu bằng	ZF=1
	nhảy nếu bằng 0	
JNE/JNZ	nhảy nếu không bằng	ZF=0
	nhảy nếu khác 0	
JC	nhảy nếu có nhớ	CF=1
JNC	nhảy nếu không nhớ	CF=0
JO	nhảy nếu nếu tràn	OF=1
JNO	nhảy nếu không tràn	OF=0
JS	nhảy nếu dấu âm	SF=1
JNS	nhảy nếu dấu dương	SF=0
JP/JPE	nhảy nếu cờ chẵn	PF=1
JNP/JPO	nhảy nếu cờ lẻ	PF=0

ệnh CMP.

Các điều kiện nhảy thường được cung cấp bởi lệnh CMP (compare). Nó có dạng sau:

```
CMP    đích,nguồn
```

Lệnh này so sánh toán tử đích với toán tử nguồn bằng cách lấy toán tử đích trừ đi toán tử nguồn. Kết quả không được lưu lại nhưng các cờ bị ảnh hưởng. Các toán hạng của lệnh CMP không thể cùng là các ô nhớ. Toán hạng đích không được phép là hằng số. Chú ý: CMP giống hệt như SUB ngoại trừ việc toán hạng đích không bị thay đổi.

Ví dụ. Giả thiết chương trình chứa hai dòng lệnh sau đây:

```
CPM    AX,BX
```

Ví dụ 6.2. Thay số trong AX bằng giá trị tuyệt đối của nó.

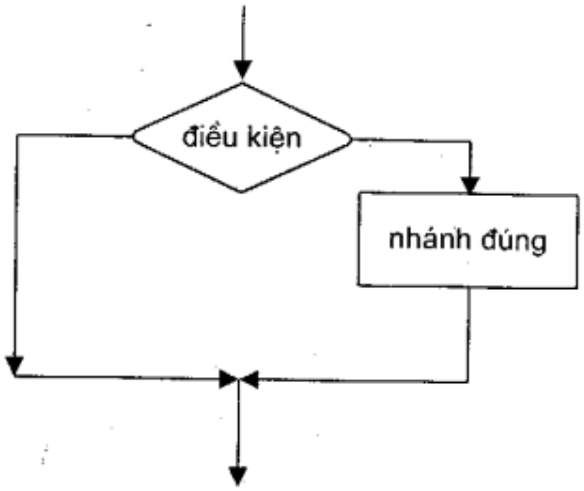
Trả lời : Một thuật toán với mã lệnh giả:

```
IF  AX>0
THEN
     thay AX bằng -AX
END_IF
```


Nó có thể được mã hoá như sau:

```
; if  AX<0
    CMP    AX,0          ; AX<0 ?
    JNL    END_IF        ; không, thoát ra.
; then
    NEG    AX            ; đúng, đổi dấu
END IF:
```

Hình 6.2. IF THEN



Ví dụ 6.3. Giả sử AL và BL chứa các ký tự ASCII mở rộng. Hãy hiển thị ký tự đứng trước trong bảng mã.

Trả lời :

```
IF  AL<=BL
    THEN
        hiển thị ký tự trong AL
ELSE
        hiển thị ký tự trong BL
END_IF
```

Ta có thể mã hoá nó như sau:

```
        MOV    AH,2          ;chuẩn bị hiển thị
;if  AL<=BL
        CMP    AL,BL         ; AL<=BL  ?
        JNBE   ELSE_         ;không, hiển thị ký tự trong BL
;then                        ;AL<=BL
        MOV    DL,AL         ;chuyển ký tự vào DL.để hiển thị
        JMP    DISPLAY       ;tới DISPLAY
ELSE_:                       ; BL < AL
        MOV    DL,BL         ;chuyển ký tự vào DL để hiển thị
DISPLAY:
        INT    21h           ; hiển thị nó.
;END_IF
```

Chú ý: Ta dùng nhãn ELSE_ vì ELSE là từ dành riêng.

Ví dụ 6.4. Nếu AX chứa một số âm, hãy nhập -1 vào BX, nếu AX chứa 0, cho BX bằng 0, nếu AX dương đổi BX thành 1.

Lời giải:

```
CASE AX
    <0: gán BX bằng -1
    =0: gán BX bằng 0
    >0: gán BX bằng 1
END_CASE
```

Ta có thể mã hoá như sau:

```
;case AX
        CMP     AX,0            ; kiểm tra AX
        JL      NEGATIVE        ; AX<0
        JE      ZERO            ; AX=0
        JG      POSITIVE        ; AX>0
NEGATIVE:
        MOV     BX,-1           ; nhập -1 vào BX
        JMP     END_CASE        ; rồi thoát
ZERO:
        MOV     BX,0            ; nhập 0 vào BX
        JMP     END_CASE        ; rồi thoát
POSITIVE:
        MOV     BX,1            ; nhập 1 vào BX
END_CASE:
```

Các điều kiện AND.

Điều kiện AND chỉ đúng khi cả hai điều kiện: điều_kiện_1 và điều_kiện_2 cùng đúng. Ngược lại nếu một trong chúng sai, điều kiện AND cũng sẽ sai.

Ví dụ 6.6. Đọc một ký tự. Nếu là chữ hoa thì hiển thị nó.

Lời giải:

```
Đọc một ký tự ( vào DL)
IF  ( 'A' <= ký_tự ) và ( ký_tự <= 'Z' )
THEN
    hiển thị ký tự
END_IF
```

Để mã hoá, đầu tiên chúng ta kiểm tra xem ký tự trong AL có đứng sau 'A' trong bảng mã hay không, nếu sai ta có kết thúc. Nếu đúng, trước khi hiển thị ký tự ta vẫn còn phải kiểm tra ký tự có đứng trước 'Z' hay không. Sau đây là mã lệnh:

```
;đọc một ký tự
        MOV    AH,1          ; chuẩn bị đọc
        INT    21h           ; ký tự vào AL
;if  ( 'A' <= ký_tự ) và ( ký_tự <= 'Z' )
        CMP    AL,'A'        ; ký_tự >='A' ?
        JNGE   END_IF        ; không, thoát ra
        CMP    AL,'Z'        ; kýtự <='Z' ?
        JNLE   END_IF        ; không, thoát ra
; then hiển thị ký tự
        MOV    DL,AL         ; lấy ký tự
        MOV    AH,2          ; chuẩn bị hiển thị
        INT    21h           ; hiển thị ký tự
END_IF:
```

Các điều kiện AND.

Điều kiện AND chỉ đúng khi cả hai điều kiện: điều_kiện_1 và điều_kiện_2 cùng đúng. Ngược lại nếu một trong chúng sai, điều kiện AND cũng sẽ sai.

Ví dụ 6.6. Đọc một ký tự. Nếu là chữ hoa thì hiển thị nó.

Lời giải:

```
Đọc một ký tự ( vào DL)
IF  ( 'A' <= ký_tự ) và ( ký_tự <= 'Z' )
THEN
    hiển thị ký tự
END_IF
```

Để mã hoá, đầu tiên chúng ta kiểm tra xem ký tự trong AL có đứng sau 'A' trong bảng mã hay không, nếu sai ta có kết thúc. Nếu đúng, trước khi hiển thị ký tự ta vẫn còn phải kiểm tra ký tự có đứng trước 'Z' hay không. Sau đây là mã lệnh:

```
;đọc một ký tự
        MOV    AH,1          ; chuẩn bị đọc
        INT    21h           ; ký tự vào AL
;if  ( 'A' <= ký_tự ) và ( ký_tự <= 'Z' )
        CMP    AL,'A'        ; ký_tự >='A' ?
        JNGE   END_IF        ; không, thoát ra
        CMP    AL,'Z'        ; kýtự <='Z' ?
        JNLE   END_IF        ; không, thoát ra
; then hiển thị ký tự
        MOV    DL,AL         ; lấy ký tự
        MOV    AH,2          ; chuẩn bị hiển thị
        INT    21h           ; hiển thị ký tự
END_IF:
```

Để mã hoá, đầu tiên chúng ta kiểm tra ký_tự = 'y' ?. Nếu thoả mãn, điều kiện OR đúng và chúng ta có thể thực hiện dòng lệnh THEN. Ngược lại vẫn cơ hội để điều kiện OR đúng, đó là khi ký_tự bằng 'Y', và dòng lệnh THEN được thi hành. Nếu điều này vẫn sai, điều kiện OR là sai và chúng ta sẽ thực hiện dòng lệnh ELSE. Sau đây là mã lệnh:

```
;đọc một ký tự
        MOV    AH,1          ; chuẩn bị đọc
        INT    21h           ; ký tự trong AL
; if   ( ký_tự = 'y' ) hoặc ( ký_tự ='Y' )
        CMP    AL,'y'        ; ký_tự = 'y' ?
        JE     THEN          ; đúng, chuyển đến hiển thị ký tự
        CMP    AL,'Y'        ; ký_tự = 'Y'
        JE     THEN          ; đúng, chuyển đến hiển thị ký tự
        JMP    ELSE_         ; sai, kết thúc
THEN:
        MOV    AH,2          ; chuẩn bị hiển thị
        MOV    DL,AL         ; lấy ký tự
        INT    21h           ; hiển thị nó
        JMP    END_IF        ; và thoát ra
ELSE_:
        MOV    AH,4Ch        ;
```



```
        INT    21h           ; trở về DOS
END_IF:
```

Ví dụ 6.8. Viết một vòng lặp điều khiển bằng biến đếm hiển thị một dòng 80 dấu sao.

Lời giải:

```
FOR  80 times DO
    hiển thị '*'
END_FOR
```

Mã lệnh là:

```
        MOV   CX,80         ; số các dấu sao được hiển thị
        MOV   AH,2          ; hàm hiển thị ký tự
        MOV   DL,'*'        ; ký tự hiển thị
TOP:
        INT   21h           ; hiển thị một dấu sao
        LOOP  TOP           ; lặp lại 80 lần
```

Ví dụ 6.9. Viết các lệnh để đếm số ký tự trong một dòng.

Lời giải:

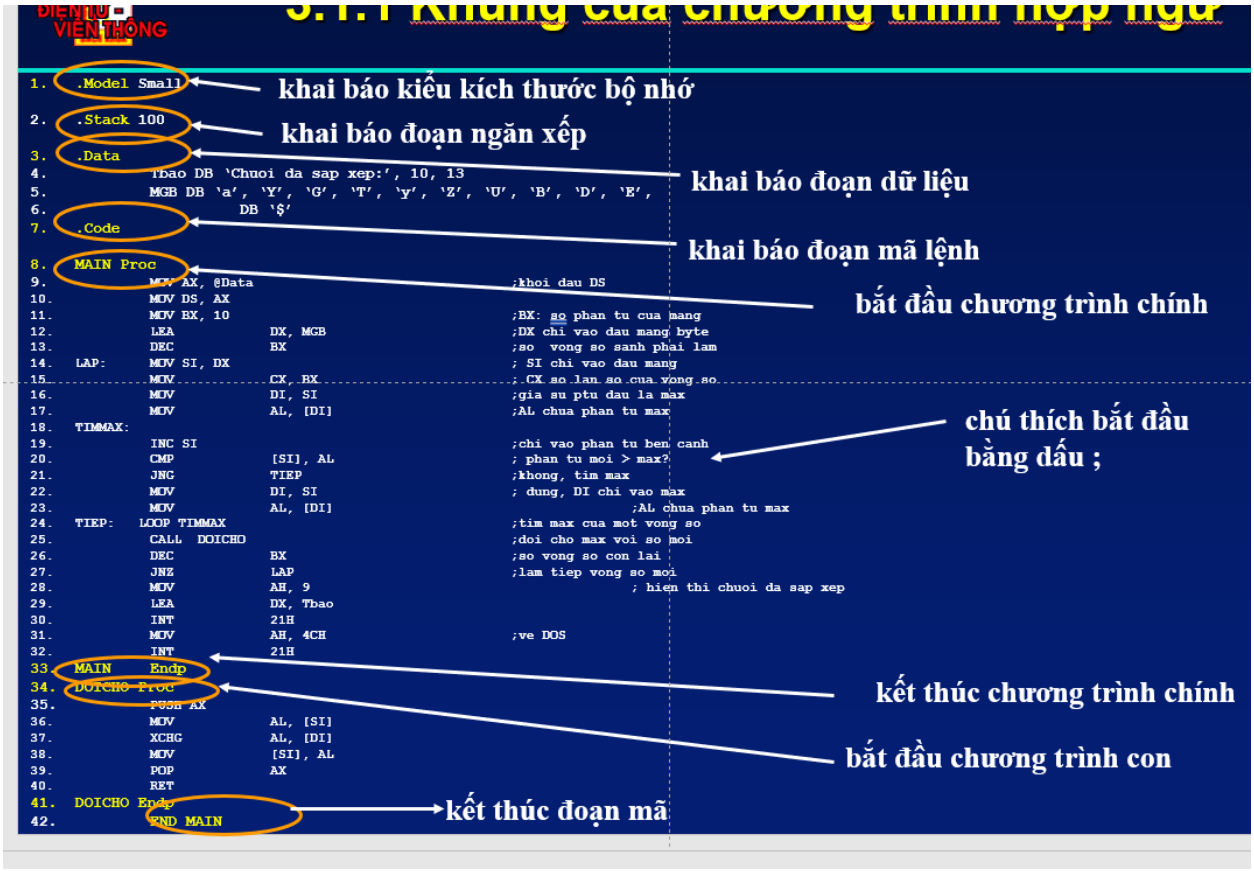
```
khởi tạo bộ đếm bằng 0,
đọc một kỷ tự
WHILE  ký tự <> ký tự về đầu dòng  DO
    đếm = đếm + 1
    đọc một ký tự
END_WHILE
```

Các lệnh là:

```
        MOV    DX,0              ; DX đếm số ký tự
        MOV    AH,1              ; chuẩn bị đọc
```



```
        INT    21h               ; ký tự trong AL
WHILE_:
        CMP    AL,0Dh            ; CR ?
        JE     END_WHILE         ; đúng, thoát ra
        INC    DX                ; không phải CR, tăng bộ đếm
        INT    21h               ; đọc một ký tự
        JMP    WHILE_            ; lặp lại
END_WHILE:
```


3.1.1 Khung của chương trình hợp ngữ
khai báo kiểu kích thước bộ nhớ
khai báo đoạn ngăn xếp
khai báo đoạn dữ liệu
khai báo đoạn mã lệnh
bắt đầu chương trình chính
chú thích bắt đầu bằng dấu ;
kết thúc chương trình chính
bắt đầu chương trình con
kết thúc đoạn mã
1. .Model Small
2. .Stack 100
3. .Data
4. Tbao DB 'Chuoi da sap xep:', 10, 13
5. MGB DB 'a', 'Y', 'G', 'T', 'y', 'Z', 'U', 'B', 'D', 'E',
6. DB '$'
7. .Code
8. MAIN Proc
9. MOV AX, @Data ;khoi dau DS
10. MOV DS, AX
11. MOV BX, 10 ;BX: so phan tu cua mang
12. LEA DX, MGB ;DX chi vao dau mang byte
13. DEC BX ;so vong so sanh phai lam
14. LAP: MOV SI, DX ; SI chi vao dau mang
15. MOV CX, BX ; CX so lan so cua vong so
16. MOV DI, SI ;gia su ptu dau la max
17. MOV AL, [DI] ;AL chua phan tu max
18. TIMMAX:
19. INC SI ;chi vao phan tu ben canh
20. CMP [SI], AL ; phan tu moi > max?
21. JNG TIEP ;khong, tim max
22. MOV DI, SI ; dung, DI chi vao max
23. MOV AL, [DI] ;AL chua phan tu max
24. TIEP: LOOP TIMMAX ;tim max cua mot vong so
25. CALL DOICHO ;doi cho max voi so moi
26. DEC BX ;so vong so con lai
27. JNZ LAP ;lam tiep vong so moi
28. MOV AH, 9 ; hien thi chuoi da sap xep
29. LEA DX, Tbao
30. INT 21H
31. MOV AH, 4CH ;ve DOS
32. INT 21H
33. MAIN Endp
34. DOICHO Proc
35. PUSH AX
36. MOV AL, [SI]
37. XCHG AL, [DI]
38. MOV [SI], AL
39. POP AX
40. RET
41. DOICHO Endp
42. END MAIN